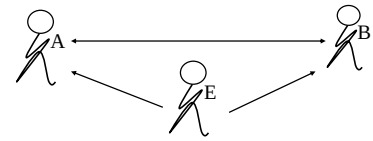
主要内容

- 身份认证（续）
- 密钥管理技术
  - 基于对称加密的密钥管理
    - kerberos的基本原理
  - 基于非对称加密的密钥管理
    - PKI基本原理
    - PKI的应用

远程用户认证的威胁：重放攻击

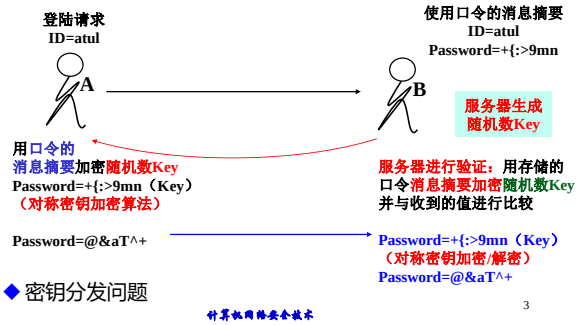
重放攻击(replay attacks)



- 偷听者可以记录下当前的通信流量，以后在适当的时候重发给通信的某一方，达到欺骗的目的
- 对策
  - 保证通信的唯一性
  - 增加时间戳

远程用户认证：增加随机性

- 使用随机数（“挑战” 码）



- 密钥分发问题

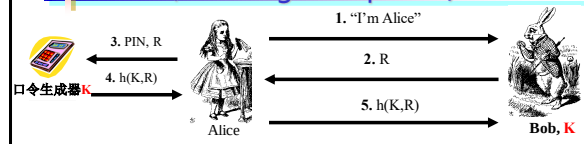
认证过程说明

1. 客户向认证服务器发出请求，要求进行身份认证；
2. 认证服务器从用户数据库中查询用户是否是合法的用户，若不是，则不做进一步处理；
3. 认证服务器内部产生一个随机数Key，作为“挑战”码，发送给客户；
4. 客户将用户名字和随机数合并，使用单向Hash函数（例如MD5算法）或对称密钥加密算法生成一个字节串作为应答；
5. 认证服务器将应答串与自己的计算结果比较，若二者相同，则通过一次认证；否则，认证失败；
6. 认证服务器通知客户认证成功或失败。

你拥有的身份证明

- 你拥有什么?
- 例如...
  - 汽车钥匙
  - 笔记本电脑（或 MAC地址）
  - 银行ATM卡, 智能卡等
- 鉴别令牌（口令生成器）
  - 处理器、LCD电池、小键盘、实时时钟
  - 产生随机数（加密密钥）

令牌协议：挑战-响应协议 (Challenge-Response)



- Alice从 Bob处接收随机挑战值 "challenge" R，并将PIN和R输入到口令生成器中
- 口令生成器将对称密钥 K和 挑战值R进行散列计算（如HMAC-MD5），生成响应值 "response"
- Alice 向Bob发送响应值 h(K,R)
- Bob验证响应值 response（使用R和密钥K进行HMAC-MD5运算）
- 注意: Alice 拥有口令生成器，并知道 PIN码 ---*双因素认证*

对称密钥K的分发和访问?

双因素认证

- 需要多个认证要素
  - 你知道什么?
  - 你拥有什么?
  - 你是谁? （生物特征）
- 例如
  - ATM卡: 卡片和PIN码
  - 信用卡：卡片和签名
  - 口令生成器: 设备和 PIN码

生物特征技术

- "你是谁？"
- 生物特征
  - 指纹
  - 手写签名
  - 面部识别
  - 语音识别
  - 步态识别
  - 气味识别
  - 等等

生物特征技术的特点

- 相对于口令认证的优势
  - 难以伪造
- 但是。。。
  - 攻击者复制指纹、虹膜图片等
  - 破坏对比软件，干扰识别行为等
- 被破解的口令可以被替换，但如何撤销被破解的生物特征?
- 生物特征识别无法做到万无一失
- 隐私问题
- 未来发展趋势?

单点登录（Single Sign-on）

- 反复输入口令费时费力
  - 用户能否只认证一次
  - 获得凭证“Credentials”后能够自动完成后续的其他认证
  - 即后续的认证对用户是透明的
- Kerberos --- 单点登录协议
- Internet上的单点登录?
  - Microsoft: **Passport**，其他: **Liberty Alliance**
  - 使用安全断言标记语言SAML (Security Assertion Markup Language)实现
  - CAS : Yale 大学发起的一个开源项目

密钥管理技术

（**Internet认证应用**）

基于对称加密的密钥分配

- 对称加密
  - 共享密钥，安全性
- 如何分发密钥?
  - 物理方法传递?
  - 用旧密钥加密新密钥?
  - 基于第三方的加密链路
- 密钥分发中心（KDC）
  - 会话密钥：逻辑连接，一次性
  - 永久密钥：用于分发会话密钥
- 实现： kerberos

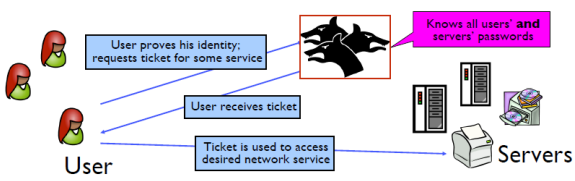
Kerberos的动机

- 使用公钥加密技术的认证
  - N 个用户 ⇒ N 个密钥对
- 使用对称密钥加密技术的认证
  - N 个用户需要O (N^2) 对密钥
- 对称密钥加密方案不具有可扩展性
- Kerberos基于对称密钥技术，但对于N个用户只需要N个密钥
  - 安全性取决于可信第三方（Trusted Third Party，TTP)

Kerberos认证协议

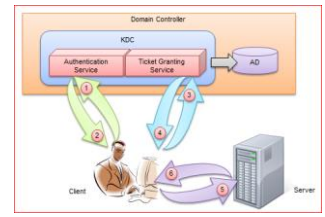
- 历史
  - 20世纪80年代中期，是**MIT的雅典娜工程**（Project Ather http://web.mit.edu/kerberos/
  - 基于Needham-Schroeder协议的认证系统
  - 版本
    - 前三个版本仅用于内部，第四版得到了广泛的应用
    - 第五版于1989年开始设计
      - **RFC 1510，(1993年)，2005年由RFC 4120取代，标准Kerberos**
- 支持的密码算法：SHA-1, MD5, DES, 3DES, AES
- 服务：认证服务，数据完整性、保密性
- 广泛应用：开源工具；商业产品；单点登录sso
  - Windows 2000/XP/Server 2003/Vista，用于域环境下的认证
  - Apple's Mac OS X clients and servers
  - Apache HTTP Server, Eudora, NFS, OpenSSH, rcp (remote copy), rsh, X window system

Kerberos认证协议



Windows的认证协议

- Windows认证协议
  - NTLM（NT LAN Manager）：主要应用于用于Windows NT 和 Windows 2000 Server（or Later） 工作组环境
  - Kerberos：主要应用于Windows 2000 Server（or Later）域（Domain）环境。



Kerberos认证的三方

- 整个认证过程涉及到三方
  - 客户端
  - 服务端
  - KDC（Key Distribution Center）
- 在Windows域环境中，KDC的角色由DC（Domain Controller）来担当。
- 应用场景
  - 当用户采用域帐号登录到某台主机，并远程访问处于相同域中另一台主机时，如何对访问者和被访问者进行身份验证（双向认证）？

需要解决的问题

- 在一个开放的分布式环境中，存在以下威胁：
  用户冒充别人的身份访问特定应用的服务器
  - 用户伪造服务器地址
  - 用户窃听信息，进行重放攻击
- 后果
  - 一个非授权用户可能获得服务和数据

Kerberos：一种认证协议

- 特点
  - 可伸缩性——可适用于分布式网络环境
  - 依赖于可信第三方（Trusted Third Party，TTP）
  - 基于口令的认证协议
  - 利用对称密码技术建立起来的认证协议
  - 环境特点
    - 双向：用户与服务器之间的认证（User-to-server authentication）
    - 密钥存储方案
  - 适用于较小规模应用场景,如局域网或公司内部网络

Kerberos的概念：主体

- **Principal(安全主体)**
  - 委托人，被认证的主体，可以是用户或服务。
  - 名字(name)和口令(password)
  - 例如：
    - 用户主体名称 user@REALM
    - 服务主体名称：
      serviceclass/host_port/serviceName

Kerberos的概念：KDC

- KDC(Key distribution center) （密钥分发中心）
  - 是一个网络服务，提供ticket和临时会话密钥
  - 网络中所有客户机和服务器都信任 KDC，并使用 KDC 验证用户身份
- KDC逻辑组成
  - 认证服务（Authentication Service，AS)
    - 域内所有主密钥（master keys）
    - 用户的主密钥，与口令相关
    - 生成TGT（ticket-granting tickets）
  - 票据授予服务（Ticket Granting Service，TGS)
    - 生成两个主体之间通信的票据（tickets）
- KDC 还拥有一个密钥数据库，每个实体（用户、主机或应用服务器）与 KDC 共享一个密钥

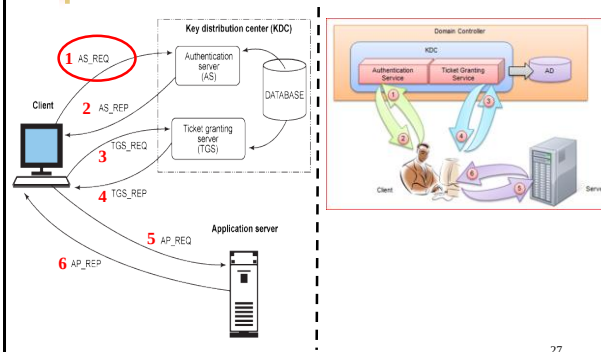
Kerberos的概念（续）

- Ticket（票据）
  - 一个记录，客户可以用它来向服务器证明自己的身份，其中包括客户的标识、会话密钥、时间戳，以及其他一些信息。Ticket 中的大多数信息都被加密，密钥为服务器的密钥
- Authenticator（认证记录）
  - 一个记录，其中包含一些最近产生的信息，产生这些信息需要用到客户和服务器之间共享的会话密钥
- Credentials（凭证）
  - 一个ticket加上一个秘密的会话密钥

Kerberos涉及的四方

- Client
  - 客户端（工作站）
- **Authentication Server(AS,认证服务器)**
  - 在登录时认证用户
- **Ticket Granting Server(TGS，票据授予服务器)**
  - 签发票据，认证身份证明
- Application server
  - 提供特定服务的服务器，网络打印、文件共享和应用程序

Kerberos的主要工作流程

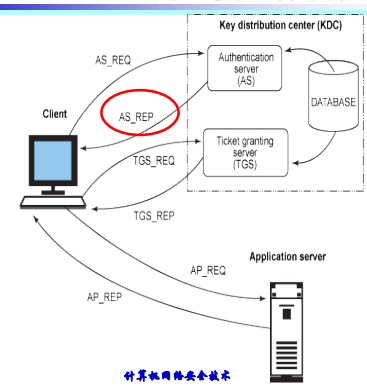


(1) AS_REQUEST

- 初始用户身份验证请求，该消息指向认证服务器（Authentication Server，AS）
- 获取TGT(ticket-granting ticket)
- K_c：客户端对用户口令执行散列运算。此散列值（即用户密钥）成为客户端和KDC共享的长期密钥（long term key）
- 用户（ client ）向AS发送消息，nonce1为随机数（时间戳）

< client id, KDC id, requested ticket expiration, nonce1 >

Kerberos的主要工作流程



(2) AS_REP

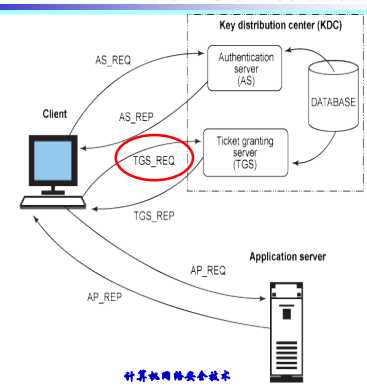
- AS 对请求的回复

< { $K_{c,TGS}$, nonce1}K_c, { $ticket_{c,tgs}$ }K_{TGS} >

- AS的响应包括两部分:
  - *Nonce1*：时间戳，防止重放攻击
  - $K_{c,TGS}$：一个新的会话密钥（用户与TGS之间的会话，使用请求用户的密钥加密）
  - $ticket_{c,tgs}$：即票据许可票据TGT，使用 K_{TGS}加密，客户无法读取或改变
  - K_{TGS}：AS与TGS的共享密钥
- Ticket说明:

$ticket_{x,y}$ = { x, y, beginning valid time, expiration time, $K_{x,y}$ }

Kerberos的主要工作流程



(3) TGS_REQUEST

◆TGS请求：客户向票据授予服务器（Ticket Granting Server，TGS）请求获得服务票据。

- ❖包含从前面的消息获取的 TGT 和一个由客户机生成的用会话密钥加密的认证记录$auth_c$ 。
- ❖$auth_c$：认证记录*authenticator*

$< \{ auth_c \} K_{c, TGS}, \{ ticket_{c, TGS} \} K_{TGS}, service, nonce2 >$

Authenticator的作用

◆认证记录*authenticator*

- ❖{$auth_c$}包括用户名（client id） 和一个新的时间戳

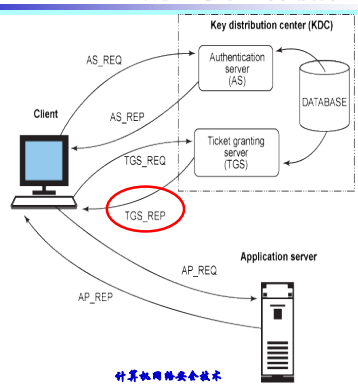
◆为了避免用户每次请求ticket时都访问AS 生成TGT（ticket-granting ticket）

- ❖TGT可以多次使用

◆Authenticator 可以防止在多次使用TGT时受到重放攻击

- ❖时间戳

Kerberos的主要工作流程



(4) TGS_REP

◆TGS_REP 是 TGS 对前面的请求的回复

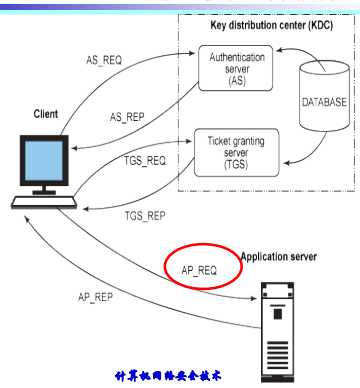
- ❖$ticket_{c, service}$：被请求的服务票据，使用该服务的私有密钥加密
- ❖$K_{c,service}$：由 TGS 生成的服务会话密钥，使用前面由 AS 生成的会话密钥加密。

$<\{K_{c,service}, nonce2\}K_{c, TGS}, \{ ticket_{c, service} \}K_{service} >$

◆注意：

- ❖用户不能读取或修改 ticket
- ❖在用户和TGS之间使用会话密钥和 *nonce2*

Kerberos的主要工作流程



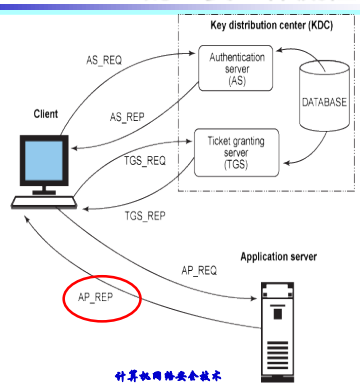
(5) AP_REQ

- AP_REQ 是客户向应用服务器发送的要求访问服务的请求。
  - $ticket_{c,service}$：通过前面的回复从 TGS 获得的服务票据
  - $Auth_c$：另一个由客户机生成的认证记录，但这一次使用服务会话密钥$K_{c,service}$加密（由 TGS 生成）。

$$<\{auth_c\}K_{c,service}, \{ticket_{c,service}\}K_{service}, request, \mathbf{nonce3}>$$

- 注意
  - 使用会话密钥
  - ticket被加密

Kerberos的主要工作流程



(6) AP_REPLY

- AP_REP 是应用服务器对客户端的回复，用于证明它就是用户所期待的服务器。通常不需要这个包。只有在需要进行双向认证时，客户机才向服务器请求这个包。

$$<\{\mathbf{nonce3}\}K_{c,service}, response>$$

- 至此，客户端和服务器之间共享一个秘密密钥
- 注意：
  - ***Nonce3*** 被加密，用户能够确认应答来自真正的应用服务器，而不是攻击者

Kerberos域

- 一个完整的Kerberos环境
  - 一台Kerberos服务器
  - 若干客户端
  - 若干应用服务器
- 要求
  - Kerberos服务器的数据库中必须存有所有参与用户的ID和经过散列函数处理过的口令。所有用户都要在Kerberos服务器上注册
  - Kerberos服务器必须与每个服务器共享一个秘密密钥。所有服务器都要在Kerberos服务器上注册
- 这种环境被称为Kerberos域

Kerberos v5的改进

- RFC 1510，RFC 4120（新）
- 改进
  - 环境不足
    - 加密算法：增加加密标识
    - 网络协议：增加网络地址类型和长度
    - 字节排序：改为ASN.1和基本编码规则BER
    - 票据有效期：增加开始和结束时间
    - 认证转发, 域间认证
  - 技术不足
    - 双重加密，非标准DES加密模式，会话密钥，口令攻击

Kerberos的安全性

- Kerberos信任模型
  - 每个主体（principal）和KDC的通信是在利用仅双方可知的密钥构建的安全通道中进行。
  - 当主体（principal）之间需要通信的时候，它们再使用KDC生成的会话密钥。
- Kerberos协议是无状态的，因此密钥分发中心和票据授予服务并没记录以前的交互信息。

Kerberos的安全性（续1）

- KDC的数据库
  - KDC持有某个域中所有主体的密码，因此 KDC 数据库的泄漏危及整个系统
  - 运行 KDC 的主机原则上不应该再运行其他服务，以最小化泄漏的风险。
- Kerberos 并不能解决密码猜测（口令攻击）和拒绝服务（DoS）等攻击
  - Kerberos 使用一个主体的密钥（加密密钥）作为它的身份的主要证明。如果一个用户的 Kerberos 密码被攻击者窃取，攻击者就能够模拟该用户。

Kerberos的安全性（续2）

- 单点故障
  - 需要中心服务器的持续响应。当Kerberos服务结束前，没有人可以连接到服务器。
- Kerberos要求参与通信的主机的时钟同步
  - 票据具有一定有效期，因此，如果主机的时钟与Kerberos服务器的时钟不同步，认证会失败。默认设置要求时钟的时间相差不超过10分钟。
- 管理协议并没有标准化，在服务器实现工具中有一些差别。

基于非对称加密的密钥分配

PKI的动机

- 公钥技术
  - 1976年，Diffie和Hellman提出公钥密码思想
    - 非对称，公钥和私钥
  - 提供**数字签名**功能，实现**不可否认服务**
- 密钥分发
  - 公钥分发
  - 使用公钥加密分发私钥

公钥证书

- 公钥和身份如何建立联系
  - 为什么要相信这是某个人的公钥
  - 公钥如何管理?
- 方案：引入证书(certificate)
  - 通过证书把公钥和身份关联起来
- 公钥证书组成
  - 公钥，用户ID，可信第三方（认证中心CA）签名

证书和身份证

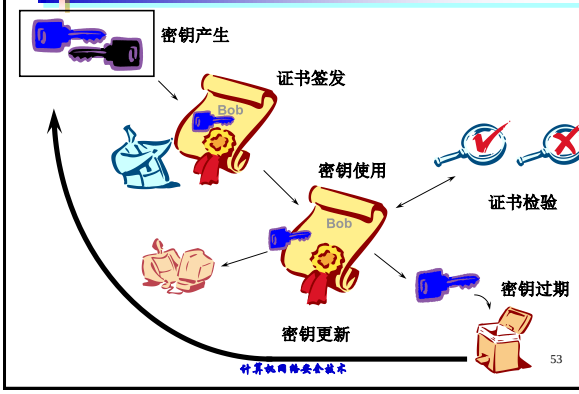
◆证书的主要内容

持有者（Subject）标识
序列号
公钥（n,e）
有效期
签发者（Issuer）标识
CA的数字签名

身份证主要内容

姓名
身份证号码
照片
有效期
签发单位
签发单位盖章、防伪标志

密钥生命周期



PKI (Public Key Infrastructure)

◆公钥基础设施 PKI（RFC4949）
- ❖用公钥原理和技术实施和提供安全服务的具有普适性的安全基础设施
- ❖用户可利用PKI平台提供的服务进行安全的电子交易，通信和互联网上的各种活动
- ❖PKI是创建、颁发、管理、注销公钥证书所涉及到的所有软件、硬件的集合体

◆目标：安全、方便和高效获取公钥

◆其核心元素是数字证书，核心执行者是CA认证机构

一个完整的PKI组成

◆ 认证机构（CA）
- ❖ 证书的签发机构，是PKI的核心，是PKI应用中权威的、可信任的、公正的第三方机构

◆ 证书库
- ❖ 是证书的集中存放地，提供公众查询

◆ 密钥备份及恢复系统
- ❖ 对用户的解密密钥进行备份，当丢失时进行恢复，而签名密钥不能备份和恢复

◆ 证书作废处理系统
- ❖ 证书由于某种原因需要作废、终止使用，这将通过证书作废列表CRL来完成

◆ PKI应用接口系统
- ❖ 是为各种应用提供安全、一致、可信任的方式与PKI交互，确保所建立起来的网络环境安全可信，并降低管理成本

PKI提供的基本服务

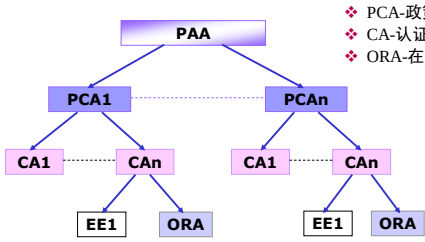
- 认证
  - 采用数字签名技术，签名作用于相应的数据之上
    - 被认证的数据 —— 数据源认证服务
    - 用户发送的远程请求 —— 身份认证服务
- 完整性
  - PKI采用了两种技术
    - 数字签名：既可以是实体认证，也可以是数据完整性
    - MAC(消息认证码)：如DES-CBC-MAC等
- 保密性
  - 用公钥分发随机密钥，然后用随机密钥对数据加密
- 不可否认
  - 发送方的不可否认 —— 数字签名
  - 接收方的不可否认 —— 收条 + 数字签名

PKI的应用考虑

- 提供基本功能的同时，还应该考虑
  - 性能
    - 尽量少用公钥加解密操作，在实用中，往往结合对称密码技术，避免对大量数据作加解密操作
    - 除非需要数据来源认证才使用签名技术，否则就使用MAC或者HMAC实现数据完整性检验
  - 在线和离线模型
    - 签名的验证可以在离线情况下完成
    - 用公钥实现保密性也可以在离线情况下完成
    - 离线模式的问题：无法获得最新的证书注销信息
  - 证书中所支持算法的通用性
    - 在提供实际的服务之前，必须协商到一致的算法
  - 个体命名：取决于CA的命名登记管理工作

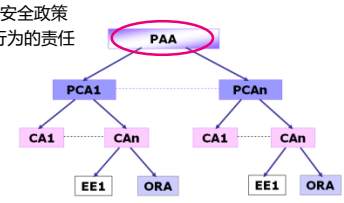
PKI 体系结构

- PAA-政策批准机构
- PCA-政策CA
- CA-认证机构
- ORA-在线证书申请



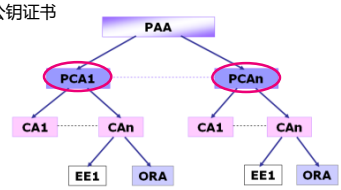
PKI体系结构

- PAA - 政策批准机构
  - 创建整个PKI系统的方针
  - 批准本PAA下属PCA的政策
  - 为下属PCA签发公钥证书
  - 建立整个PKI体系的安全政策
  - 并具有检测各PCA行为的责任



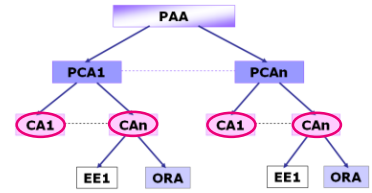
PKI体系结构

- PCA-政策CA
  - 制定本PCA下的具体政策
  - 可以是PAA政策的扩充或细化，但不能与之相背离
  - 这些政策可能包括本PCA范围内密钥的产生、长度、证书的有效期规定、CRL的处理等。
  - 并为下属CA签发公钥证书



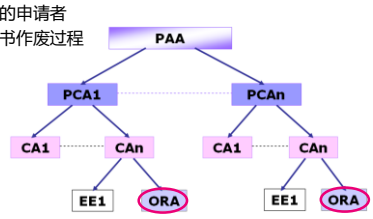
PKI体系结构

- CA-认证机构
  - 不具备或具备有限的政策制定能力
  - 按照上级PCA制定的政策，担任具体的用户公钥证书的生成和发布，或CRL生成发布职能。



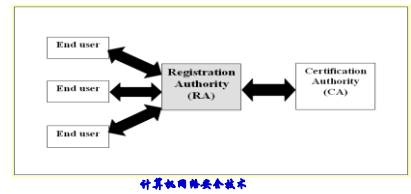
PKI体系结构

- ORA-在线证书申请
  - 进行证书申请者的身份认证
  - 向CA提交证书申请请求
  - 验证接收到的CA签发的证书
  - 并将之发放给证书的申请者
  - 必要时，还协助证书作废过程



PKI基本组件

- RA(Registration Authority)
  - 把用户的身份和它的密钥绑定起来——建立信任关系
- CA(Certificate Authority)
  - **证书签发**
- 证书库/目录
  - 保存证书，供公开访问



认证机构CA(Certificate Authority)

- 职责
  - 接受用户的请求
  - (由RA负责对用户的身份信息进行验证)
  - 用自己的私钥签发证书
  - 提供证书查询
  - 接受证书注销请求
  - 提供证书注销表
- 各个组件和功能示意图

Registration Authority - RA

- 进行用户身份信息的审核，确保其真实性
- 本区域用户身份信息管理和维护
- 数字证书的下载
- 数字证书的发放和管理
- 收集和管理申报材料和信息
- 录入身份信息
- 初步审核与提交身份信息
- 制作数字证书
- 发放数字证书

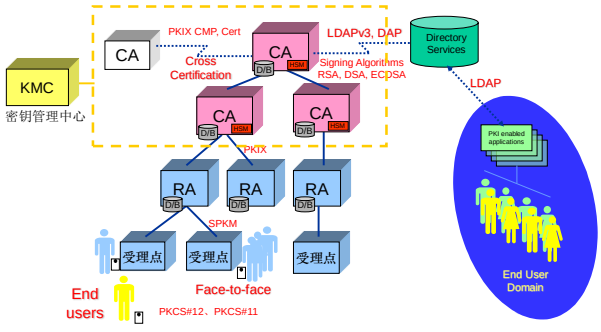
证书的签发

- CA 对用户签发证书
  - 对某个用户公钥，使用 CA 的私钥对其进行签名。这样任何人都可以用 CA 的公钥对该证书进行合法性验证。
  - 验证成功则认可该证书中所提供的用户公钥内容，实现用户公钥的安全分发。
- 两种方式
  - 由 CA 直接来生成证书（内含公钥）和对应的私钥发给用户
  - 由用户自己生成公钥和私钥，然后由 CA 来对公钥内容进行签名。

例如：用户生成公钥和私钥

- 用户自行生成一个私钥和证书申请文件（Certificate Signing Request，即 csr 文件），该文件中包括了用户对应的公钥和一些基本信息，如通用名（common name，即 cn）、组织信息、地理位置等。
- CA 对证书请求文件进行签名，生成证书文件，颁发给用户。
- 用户必须保证私钥的安全性
- 用户可以使用开源软件（如openssl等） 来生成 csr 文件和对应的私钥。
- 注意：
  - 私钥文件一旦丢失，则通过该证书中公钥加密的内容将无法被解密。

一个PKI系统的组织结构

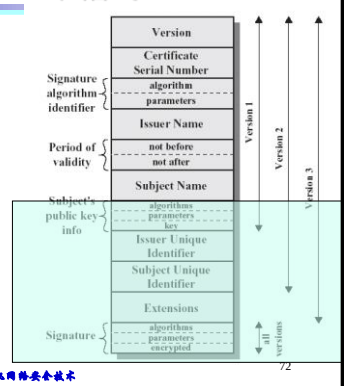


PKI中的证书

- 证书(certificate)，有时候简称为cert
- PKI适用于异构环境中，所以证书的格式在所使用的范围内必须统一
- 证书是一个机构颁发给一个安全个体的证明，所以证书的权威性取决于该机构的权威性
- 一个证书中，最重要的信息是个体名字、个体的公钥、机构的签名、算法和用途
- 最常用的证书格式为X.509 v3

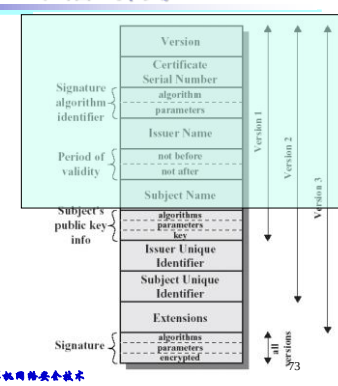
X.509证书格式

- 版本1、2、3
- 序列号
  - 在CA内部唯一
- 签名算法标识符
  - 指该证书中的签名算法
- 签发人名字
  - CA的名字
- 有效时间
  - 起始和终止时间
- 个体名字

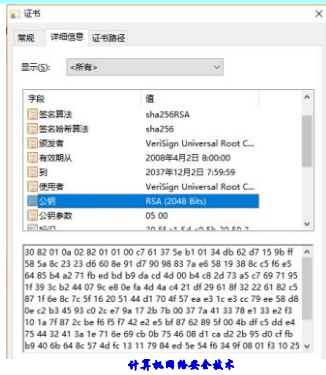


X.509证书格式(续)

- 个体的公钥信息
  - 算法
  - 参数
  - 密钥
- 签发人唯一标识符
- 个体唯一标识符
- 扩展域
- 签名



例：X.509证书



CA信任关系

- 当一个安全个体看到另一个安全个体出示的证书时，他是否信任此证书？
  - 信任难以度量，总是与风险联系在一起
- 可信CA
  - 如果一个个体假设CA能够建立并维持一个准确的“个体-公钥属性”之间的绑定，则他可以信任该CA，该CA为可信CA
  - 验证签名

CA颁发机构

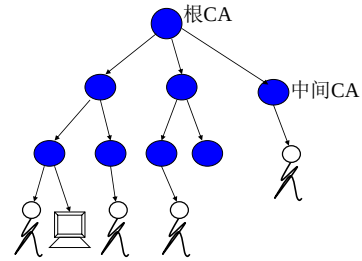
- CA认证服务
  - Verisign是第一家CA厂商，创办于1995年，在2010年以12.8亿美元卖给了赛门铁克Symantec 。
  - 受浏览器默认信任的 CA 大厂商有很多，如Symantec、Comodo、Godaddy、GolbalSign 和 Digicert。
- 我国CA安全认证系统
  - 各地区各行业自行建立CA认证机构，行业和区域性CA发展很快
  - 在面向大众的互联网数字证书如SSL证书应用领域仍然依赖于国外已经发展成熟的证书产业
    - 只有通过WebTrust国际安全审计认证的根证书才能预装到主流的浏览器中，国内仅有CNNIC CA、沃通CA、上海CA和中国金融认证中心CFCA通过了WebTrust认证，完成了在主流浏览器IE、Firefox、Chrome等根证书的嵌入。（http://www.cac.gov.cn/2015-06/02/c_1115480021.htm）

CA信任模型

- 基于层次结构的信任模型
- 交叉认证

CA层次结构

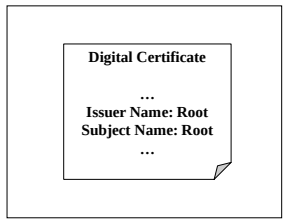
- 对于一个运行CA的大型权威机构而言，签发证书的工作不能仅仅由一个CA来完成
- 它可以建立一个CA层次结构



验证根CA

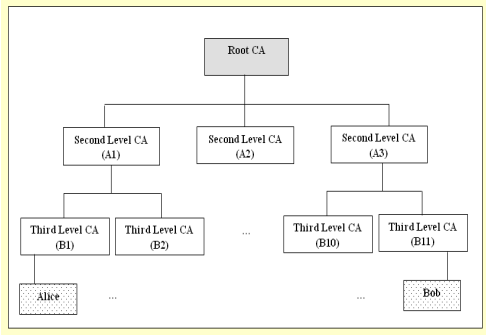
谁为根CA签名？

- **根CA签发自己的证书**
- **称为自签名证书**



CA层次结构的建立

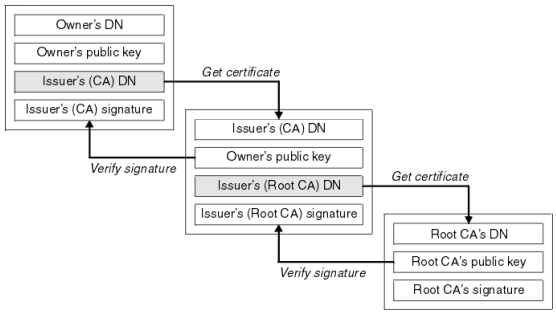
- 根CA具有一个自签名的证书
- 根CA依次对它下面的CA进行签名
- 层次结构中叶子节点上的CA用于对安全个体进行签名
- 对于个体而言，它需要信任根CA，中间的CA可以不必关心(透明的)；同时它的证书是由底层的CA签发的
- 在CA的机构中，要维护这棵树
  - 在每个节点CA上，需要保存两种cert
    (1) Forward Certificates: 其他CA发给它的certs
    (2) Reverse Certificates: 它发给其他CA的certs



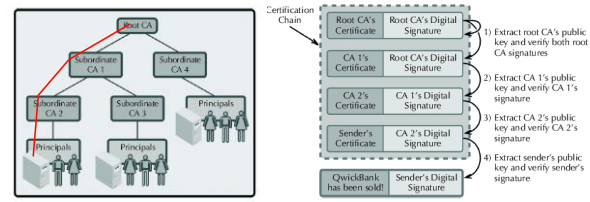
层次结构CA中证书的验证

- 假设个体A看到B的一个证书
- B的证书中含有签发该证书的CA的信息
- 沿着层次树往上找，可以构成一条证书链，直到根证书
- 验证过程：
  - 沿相反的方向，从根证书开始，依次往下验证每一个证书中的签名。其中，根证书是自签名的，用它自己的公钥进行验证
  - 一直到验证B的证书中的签名
  - 如果所有的签名验证都通过，则A可以确定所有的证书都是正确的，如果他信任根CA，则他可以相信B的证书和公钥
- 问题：证书链如何获得？

证书链的验证示例

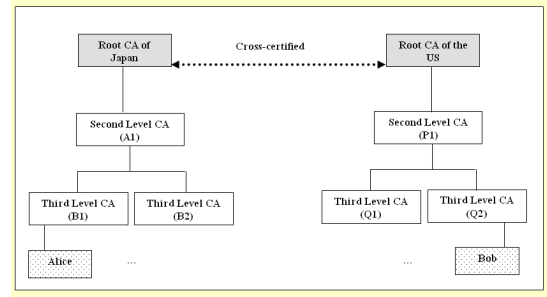


证书链的验证示例



交叉认证

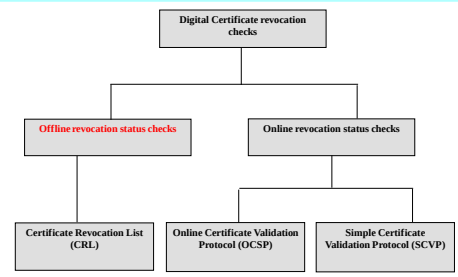
- 两个不同的CA层次结构之间可以建立信任关系
  - 单向交叉认证
    - 一个CA可以承认另一个CA在一定名字空间范围内的所有被授权签发的证书
  - 双向交叉认证
- 交叉认证可以分为
  - 域内交叉认证(同一个层次结构内部)
  - 域间交叉认证(不同的层次结构之间)



证书的注销机制

- 由于各种原因，证书需要被注销
  - 比如，私钥泄漏、密钥更换、用户变化
- PKI中注销的方法
  - 脱机证书吊销检查
    - CA维护一个CRL(Certificate Revocation List)
    - 检查CRL的URL应该内嵌在用户的证书中
  - 联机证书吊销状态检查
    - 联机证书状态协议OSCP(Online Certificate Status Protocol)
    - 简单证书检验协议SCVP(Simple Certificate Validation Protocol)

证书的注销机制



证书吊销列表
由CA定期发布

联机证书状态协议
请求-响应模式

简单证书状态协议
请求-响应模式
更详细的信息

私钥保护

方式	描述
Password	简单、常用的保护私钥的方式，用文件的方式存储在硬盘上，使用Password或用户的PIN（ *Personal Identification Number*）访问. 任何可以接触到文件的人，可通过破解口令得到私钥
PCMCIA cards	PCMCIA card 是一种芯片卡. 私钥非存储在硬盘上. 降低了被拷贝的风险. 在签名或解密的过程中，私钥需要拷贝到计算机的内存中，因此，也有可能被入侵者获得私钥
Tokens	通过 token 的加密方式存储的私钥，每一次访问使用通过token生成的一次性密码
Biometrics	访问私钥需要使用用户的生物特征，（如指纹、虹膜识别、声纹比较、人脸识别等）
Smart cards	私钥存储在智能卡中的芯片上，同时，智能卡中的芯片可以进行签名和加密操作. 私钥的操作过程不离开智能卡

与PKI有关的标准情况

- Certificates —— X.509 v.3
- PKIX group in IETF(RFC 2459)
  - 密钥管理
  - 交叉认证
- 智能卡/硬件插件 PKCS #11
- PKCS系列
  - 公钥加密标准
  - RSA公司开发
- 目录服务LDAP

PKI应用

- 基本的应用
  - 文件保护
  - E-mail
  - Web应用
- 其他
  - VPN
  - SSL/TLS
  - XML/e-business
  - WAP
  - ……

PKI实现的选择

- 建立自己的PKI
- 购买一个PKI
- 从第三方购买PKI服务
- 等待一个政府PKI
- PKI和应用联合开发，相互协作

作业

- 什么是双因素认证?
- 挑战-响应协议的基本原理?
- 口令认证中如何防止中间人攻击?
- Kerberos的系统的基本组成是什么?
- 查看你的浏览器中使用的X.509证书，并说明CA的作用。

注意：4月10日课上讨论大作业开题，
请在截止之前提交（参见课程中心“作业”）